

| | | |
|-----|-----|--|
| (2) | 102 | DECLARATIONS |
| (3) | 178 | SYSSETPFM - Initialize Page Fault Monitoring |
| (4) | 485 | PFMSPURGE - Return all process buffers to pool |
| (5) | 597 | PFMSGETBUF - Return PFM Buffer to Caller |
| (6) | 687 | ALLPMB - Allocate PMB control block and data buffers |
| (7) | 794 | PFMSMON - Resident Monitoring Code |

0000 1 .TITLE SYSSETPFM - SET PAGE FAULT MONITORING
0000 2 .IDENT 'V04-000'
0000 3 .
0000 4 .
0000 5 .*****
0000 6 .
0000 7 .* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8 .* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9 .* ALL RIGHTS RESERVED.
0000 10 .
0000 11 .* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12 .* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13 .* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14 .* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15 .* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16 .* TRANSFERRED.
0000 17 .
0000 18 .* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19 .* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20 .* CORPORATION.
0000 21 .
0000 22 .* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23 .* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24 .
0000 25 .
0000 26 .*****
0000 27 .
0000 28 .++
0000 29 .FACILITY: Measurement System Service
0000 30 .
0000 31 .ABSTRACT:
0000 32 .
0000 33 . This module enables a page fault monitoring service within the
0000 34 . operating system. On each page fault, the virtual address, the PC,
0000 35 . and the process CPU time are saved in a buffer to be output by either
0000 36 . a cooperating subprocess or an image-based AST routine.
0000 37 .
0000 38 .ENVIRONMENT: Kernel Mode
0000 39 .
0000 40 .AUTHOR: Henry M. Levy , CREATION DATE: 7-May-1977
0000 41 .
0000 42 .MODIFIED BY:
0000 43 .
0000 44 . V03-007 SSA0026 Stan Amway 9-Jul-1984
0000 45 . In PFMSMON, raise IPL to IPLS HWCLK when
0000 46 . doing CPU time reference update.
0000 47 .
0000 48 . V03-006 SSA0025 Stan Amway 25-Jun-1984
0000 49 . Add global symbol PFMSC_BUFCNT for use by PCA.
0000 50 .
0000 51 . V03-005 SSA0019 Stan Amway 12-Mar-1984
0000 52 . Prevent subprocess running PFMFILWRT from deallocating PMB.
0000 53 . Add access mode checking for subfunction and stop requests.
0000 54 .
0000 55 . V03-004 SSA0013 Stan Amway 28-Feb-1984
0000 56 . Properly handle monitoring termination by
0000 57 . dequeuing AST block from PCB.

0000 58 : Ensure that PFMSPURGE will be invoked if service
0000 59 : is called with a stop monitoring request (notably
0000 60 : from SYSRUNDWN) and a PMB is still allocated.
0000 61 : Inhibit buffer flush if buffer contains no records.
0000 62 : Fix bug that inhibited the entry of a CPU timestamp
0000 63 : into the first page fault buffer.
0000 64 : Move PFMSGETBUF and ALLPMB routines to paged
0000 65 : PSECT.
0000 66 :
0000 67 : V03-003 SSA0007 Stan Amway 2-Feb-1984
0000 68 : Removed restriction of 1 process per group.
0000 69 : Added AST interlock flag to keep the number of
0000 70 : ASTs delivered to a minimum.
0000 71 : Track changes in buffer format.
0000 72 :
0000 73 : V03-002 SSA0004 Stan Amway 12-Dec-1983
0000 74 : Extensive changes to add support for the
0000 75 : Performance & Coverage Analyzer (PCA) being done by
0000 76 : the VMS Debug group.
0000 77 :
0000 78 : Changes include:
0000 79 : Removed use of PMB list. Use PCB\$L PMB (new) instead.
0000 80 : Timestamping fault with process CPU time.
0000 81 : Adding support for image-based buffer handling.
0000 82 : Cleanup of error handling.
0000 83 : Optimizing main code paths for speed.
0000 84 :
0000 85 : V03-001 CWH1002 CW Hobbs 1-Mar-1983
0000 86 : Convert to use extended pids.
0000 87 :
0000 88 : 02 RIH0033 R. I. HUSTVEDT 16-OCT-1979
0000 89 : CHANGE PCB\$W_BYTCNT TO JIB\$L_BYTCNT.
0000 90 :
0000 91 : 03 BLS0001 B. L. SCHREIBER 28-NOV-1979
0000 92 : CORRECT PAGE FAULT ERROR
0000 93 :
0000 94 : 04 BLS0002 B.L. SCHREIBER 28-JAN-1980
0000 95 : CORRECT ERROR IN PURGE ROUTINE.
0000 96 :
0000 97 : 05 BLS0003 B.L. SCHREIBER 30-JAN-1980
0000 98 : MORE ERRORS IN PURGE ROUTINE
0000 99 :
0000 100 :--

```

0000 102 .SBTTL DECLARATIONS
0000 103 :
0000 104 : INCLUDE FILES:
0000 105 :
0000 106 :
0000 107 :
0000 108 : MACROS:
0000 109 :
0000 110 :
0000 111 $ACBDEF : define AST control block
0000 112 $DYNDEF : define dynamic structure types
0000 113 $IODEF : define I/O function codes
0000 114 $IPLDEF : define interrupt priority levels
0000 115 $JIBDEF : define job information block
0000 116 $PCBDEF : define process control block
0000 117 $PHDDEF : define process header
0000 118 $PFBDEF : define PFM buffer layout
0000 119 $PMBDEF : define PFM control block
0000 120 $PQLDEF : define process quota codes
0000 121 $PRDEF : define processor registers
0000 122 $PRIDEF : define priority increment classes
0000 123 $PSLDEF : define PSL fields
0000 124 $SGNDEF : define system parameters
0000 125 $SSDEF : define service status codes
0000 126 :
0000 127 .MACRO $QUOTA NAME=LISTEND,VALUE=0
0000 128 .BYTE PQLS 'NAME
0000 129 .LONG VALUE
0000 130 .ENDM $QUOTA
0000 131 :
0000 132 :
0000 133 : EQUATED SYMBOLS:
0000 134 :
0000 135 :
00000004 0000 136 PFMFLG = 4 : Argument list offsets
00000008 0000 137 ASTADR = 8 : Function/subfunction flags
0000000C 0000 138 ASTPRM = 12 : AST Routine Address
00000010 0000 139 ACMODE = 16 : AST Parameter
00000005 0000 140 :
00000005 0000 141 BUFCNT = 5 : AST Mode
00000005 0000 142 PFMSC BUFCNT == BUFCNT : number of buffers to allocate
0000020C 0000 143 BUFSIZ = PFBSC LENGTH : global definition of BUFCNT
0000003F 0000 144 MAXREC = <<BUF$IZ-PFB$B_BUFFER>/<2*4>> : size of buffers
00000A7C 0000 145 QUOTA CHARGE = <<BUFCNT*BUFSIZ> + PMBSC_LENGTH>> : max records per buffer
0000003C 0000 146 FAULTVA = <4*7>+4+<7*4> : Amt to charge process byte quota
00000040 0000 147 FAULTPC = <4*7>+8+<7*4> : offset to va of faulting instruction
00000000 0000 148 :
00000000 0000 149 :
00000000 0000 150 : OWN STORAGE:
00000000 0000 151 :
00000000 0000 152 :
00000000 0000 153 .PSECT YEXEPAGED,LONG
00000000 0000 154 :
00000000 0000 155 :
00000000 0000 156 : Data for creation of subprocess to output filled
00000000 0000 157 : buffers to disk file.
00000000 0000 158 :

```

| | | | |
|--|---|---|--|
| 50 3A 4D 45 54 53 59 53 24 53 59 53 45 58 45 2E 54 52 57 4C 49 46 4D 46 | 00000008: 00000008: 00000018: 00000008: 00000008: 0008 10\$: 0014 0020 0020 0020 0025 002A 002F 0034 0039 003E 0043 0043 004F 0052 0052 | 0000 159 0000 160 PFMFILWRT: 0000 161 .LONG 0004 162 .LONG 0008 163 10\$: 0014 0020 164 20\$: 0020 165 PFMQUOTA: 0020 166 \$QUOTA CPULM,0 0025 167 \$QUOTA BYTLM,1024 002A 168 \$QUOTA FILLM,1 002F 169 \$QUOTA PGFLQUOTA,256 0034 170 \$QUOTA PRCLM,0 0039 171 \$QUOTA TQUELM,1 003E 172 \$QUOTA LISTEND 0043 173 FILWRT: 0043 174 .ASCII /PFMSUB_< EPID >/ 004F 0052 175 FILWRTPRV: 0052 176 .LONG -1,-1 | 20\$-10\$ 10\$ /SYSSYSTEM:PFMFILWRT.EXE/ ; subprocess image name descriptor ; subprocess quota name ; infinite CPU time ; byte limit for buffered I/O ; open file count limit ; paging file quota ; no subprocesses ; timer queue entry ; end of list ; subprocess process name ; subprocess privilege vector ; all privileges |
|--|---|---|--|

005A 178 .SBTTL SYSSETPFM - Initialize Page Fault Monitoring
005A 179 :++
005A 180 : FUNCTIONAL DESCRIPTION:
005A 181 :
005A 182 : Page fault monitoring initialization. Buffers are allocated from
005A 183 : the nonpaged pool and queued for use by the monitor. In subprocess
005A 184 : mode, a subprocess is created which outputs buffers which have been
005A 185 : filled. In image-based mode, ASTs are delivered to signal full buffers.
005A 186 :
005A 187 : When the process calls SETPFM to turn off monitoring, all
005A 188 : buffers are returned to the system and if in subprocess mode, the
005A 189 : subprocess is deleted.
005A 190 :
005A 191 : In case of abnormal termination, the buffers are returned by SYSRUNDWN.
005A 192 :
005A 193 : CALLING SEQUENCE:
005A 194 :
005A 195 : CALLS/CALLG
005A 196 :
005A 197 : INPUT PARAMETERS:
005A 198 :
005A 199 : 4(AP) PFMFLG Function/subfunction
005A 200 : bit 0 = off/on (0/1)
005A 201 : bits 1-30 = subfunction field
005A 202 : bit 1 = Flush buffers
005A 203 : bit 31 = 0 indicates initialization call
005A 204 : 1 indicates subfunction call (if bit 0 = 1)
005A 205 :
005A 206 : 8(AP) ASTADR AST Routine Address
005A 207 : = 0 Subprocess Mode
005A 208 : <> 0 AST Routine address for image-based buffer handler
005A 209 :
005A 210 : 12(AP) ASTPRM AST parameter
005A 211 :
005A 212 : 16(AP) ALMODE Access Mode for AST delivery
005A 213 :
005A 214 :
005A 215 : IMPLICIT INPUTS:
005A 216 :
005A 217 : none
005A 218 :
005A 219 : OUTPUT PARAMETERS:
005A 220 :
005A 221 : none
005A 222 :
005A 223 : IMPLICIT OUTPUTS:
005A 224 :
005A 225 : none
005A 226 :
005A 227 : COMPLETION CODES:
005A 228 :
005A 229 : SSS_NORMAL - Success
005A 230 : SSS_EXQUOTA - A quota was exceeded while allocating buffers
005A 231 : or creating the cooperating subprocess.
005A 232 : SSS_INSFMEM - Insufficient dynamic memory was available for buffering
005A 233 : SSS_PFMBSY - Attempted to initialize page fault monitoring while
005A 234 : already active.

```

005A 235 : SSS_NOPRIV - Caller's access mode is less privileged than the mode
005A 236 : that started page fault monitoring.

005A 237 : SIDE EFFECTS:
005A 238 : none
005A 239 :
005A 240 :
005A 241 :
005A 242 ;--
005A 243 :
005A 244 :
005A 245 :
0000005A 246 .PSECT YEXEPAGED
005A 247 :
005A 248 .ENABLE LSB
005A 249 PFMBUSY:
50 0204 8F 3C 005A 250 MOVZWL #SSS_PFMBSY,RO
05 11 005F 251 BRB 10$  

50 02DC 8F 3C 0061 252 ILLSEQOP:
0066 253 MOVZWL #SSS_ILLSEQOP,RO
0066 254 10$: .DISABLE LSB
0066 255 SETIPL #0 ; Restore IPL
04 0069 256 RET ; Return w/error in R0
006A 257 :
0FFC 006A 258 .ENTRY EXESSETPFM,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
006C 259 :
006C 260 :
54 00000000'EF D0 006C 261 MOVL L^SCH$GL_CURPCB,R4 ; get our PCB address
55 00000000'9F D0 0073 262 MOVL @#CTL$GL_PHD,R5 ; and PHD address (P1 window)
007A 263 SETIPL #IPLS_ASTDEL ; Protect access to PMB
56 011C C4 D0 007D 264 MOVL PCBSL_PMB(R4),R6 ; Get address of allocated PMB (if any)
53 04 AC D0 0082 265 MOVL PFMFLG(AP),R3 ; Get copy of PFMFLG
0086 266 :
0086 267 : From this point on, IPL is at IPLS_ASTDEL and registers are as follows:
0086 268 :
0086 269 : R3 Copy of PFMFLG
0086 270 : R4 PCB Address
0086 271 : R5 PHD Address (P1 window)
0086 272 : R6 PMB Address or 0 (initialize request only)
0086 273 :
0086 274 : *** Order of following tests makes
0086 275 : implicit checks on PFM state ***
58 36 A5 00 E1 0086 276 BBC #PHDSV_PFMFLG,- ; BR if monitoring not initialized
37 53 53 E9 0088 277 PHDSW_FLAGS(R5),START
D1 18 008E 278 BLBC R3,STOP VEC
0090 279 BGEQ ILLSEQOP ; BR if termination request
0090 280 :
0090 281 : Subfunction Processing
0090 282 :
0090 283 :
0090 284 :
0090 285 SUBFUNC:
1A 10 0090 286 BSBB CHECK_ACMODE ; If caller does not have privilege,
29 50 E9 0092 287 BLBC R0,NOPRIV ; return with error status
06 53 01 E1 0095 288 BBC #1,R3,10$ ; If flush buffer request,
00000065'EF 16 0099 289 JSB FLUSH_BUFFER ; do it now
009F 290 :
009F 291 ASSUME PMBSV_MODE EQ 0 ; (R2, R4, R5 DESTROYED)

```

| | | | | | | | |
|----|------|------|------|-------|---------------|----------------------------------|---|
| 02 | 0B | A6 | 009F | 292 | ASSUME | PMB\$K_IMAGE_EQ_1 | |
| 71 | 10 | 009F | 293 | 10\$: | BLBC | PMB\$B_FLAGS(R6),20\$ | ; If image-based mode, |
| 50 | 01 | 3C | 00A3 | 294 | BSBB | SET_ASTMODE | ; update AST parameters |
| | | 04 | 00A5 | 295 | 20\$: | SETIPL #0 | ; Lower IPL |
| | | | 00AB | 296 | MOVZWL | #SSS_NORMAL,RO | ; and return with success |
| | | | 00AC | 297 | RET | | |
| | | | 00AC | 298 | | | |
| | | | 00AC | 299 | | | |
| | | | 00AC | 300 | | | |
| | | | 00AC | 301 | CHECK_ACMode: | | |
| | | 50 | D4 | 302 | CLRL | RO | |
| | | 51 | DC | 303 | MOVPSL | R1 | ; Assume function not allowed to caller |
| 02 | 16 | EF | 00B0 | 304 | EXTZV | #PSL\$V_PRVMOD,#PSLSS_PRVMOD,- | ; R1 <= PSL |
| 51 | 51 | 00B3 | 305 | | | R1,R1 | ; R1 <= caller's mode |
| 2E | A6 | 51 | 91 | 306 | CMPB | R1,PMB\$B_ACMode(R6) | ; If caller less privileged than owner, |
| 02 | | 02 | 1A | 307 | BGTRU | 10\$ | return with no access indication in RO |
| | | 50 | D6 | 308 | INCL | RO | ; else return access OK |
| | | 05 | 00BD | 309 | 10\$: | RSB | |
| | | | 00BE | 310 | | | |
| | | | 00BE | 311 | NOPRIV: | | |
| 50 | 24 | 3C | 00BE | 312 | MOVZWL | #SSS_NOPRIV,RO | ; Return w/no privilege error |
| | | | 00C1 | 313 | SETIPL | #0 | |
| | | | 00C4 | 314 | RET | | |
| | | | 00C5 | 315 | | | |
| | | 0142 | 31 | 00C5 | 316 | STOP_VEC: | |
| | | | 00C8 | 317 | BRW | STOP | |
| | | 0161 | 31 | 00C8 | 318 | PURGE_EXIT_RO_VEC: | |
| | | | 00CB | 319 | BRW | PURGE_EXIT_RO | |
| | | 015B | 31 | 00CB | 320 | PURGE_EXIT_VEC: | |
| | | | 00CE | 321 | BRW | PURGE_EXIT | |
| | | | 00CE | 322 | | | |
| | | | 00CE | 323 | CHECK_STOP: | | |
| | | 90 | E8 | 324 | BLBS | R3,ILLSEQOP | : Error if not termination request |
| | | 56 | D5 | 325 | TSTL | R6 | : If PMB doesn't exist, |
| | | 8C | 13 | 326 | BEQL | ILLSEQOP | : don't proceed |
| | | 00 | E0 | 327 | BBS | #PMB\$V_MODE,- | : If image mode, continue normally |
| EB | 0B | A6 | 00D7 | 328 | | PMB\$B_FLAGS(R6),STOP_VEC | |
| 34 | A6 | D1 | 00DA | 329 | CMPL | PMB\$L_EPID(R6),- | : Don't allow PFM rundown |
| 64 | A4 | | 00DD | 330 | | PCBSL_EPID(R4) | : if the current process is the |
| 80 | | 13 | 00DF | 331 | BEQL | ILLSEQOP | : subprocess running PFMFILEWR |
| E2 | | 11 | 00E1 | 332 | BRB | STOP_VEC | : Otherwise, join STOP monitoring code |
| | | | 00E3 | 333 | : | | |
| | | | 00E3 | 334 | : | Initialize page fault monitoring | |
| | | | 00E3 | 335 | : | | |
| | | | 00E3 | 336 | : | | |
| | | | 00E3 | 337 | START: | | |
| 01 | 53 | D1 | 00E3 | 338 | CMPL | R3,#1 | : Valid request ? |
| | E6 | 12 | 00E6 | 339 | BNEQ | CHECK_STOP | : BR if not |
| | 56 | D5 | 00E8 | 340 | TSTL | R6 | : PMB allocated ? |
| | 03 | 13 | 00EA | 341 | BEQL | 10\$ | |
| | FF6B | 31 | 00EC | 342 | BRW | PFMBUSY | |
| | 0249 | 30 | 00EF | 343 | 10\$: | BSBW | : Yes - can't proceed |
| | | | 00F2 | 344 | | ALLPMB | : Allocate PMB |
| | | | 00F5 | 345 | | | : It is now safe to lower IPL |
| DD | 50 | E9 | 00F5 | 346 | BLBC | RO,PURGE_EXIT_RO_VEC | : If PMB allocation error, quit now |
| | 51 | DC | 00F8 | 347 | MOVPSL | R1 | : Store caller's access mode |
| 02 | 16 | EF | 00FA | 348 | EXTZV | #PSL\$V_PRVMOD,#PSLSS_PRVMOD,- | |
| | 51 | | 00FD | | | R1,R1 | : as owner's access mode |

```

2E A6 51 90 00FF 349      MOVB   R1, PMBSB_ACMODE(R6)      ; for subsequent privilege checking
08 AC D5 0103 350      TSTL   ASTADR(AP)          ; Are we initializing image-based mode ?
28 13 0106 351      BEQL   SET_SUBPMODE        ; If no, setup subprocess mode
0C 10 0108 352      BSBB   SET_ASTMODE         ; Else setup image-based AST mode
00 0A 010A 353      ASSUME PMBSV_MODE EQ 0
00 0A 010A 354      ASSUME PMBSK_IMAGE EQ 1
0B A6 96 010A 355      INCB   PMBSB_FLAGS(R6)      ; Set image-based AST mode flag
00 36 00 E2 010D 356      START_OK:
50 36 A5 50 010F 357      BBSS   #PHDSV_PFMFLG_-
50 01 3C 0112 358      PHDSW_FLAGS(R5),10$    ; Enable page fault monitoring
04 0115 359      10$:  MOVZWL #SSS_NORMAL, R0  ; and return with success status
0116 360      RET
0116 361
0116 362      :
0116 363      : Setup image-based mode AST parameters
0116 364      :
0116 365
0116 366      SET_ASTMODE:
0116 367      ASSUME PMBSL_ASTPRM EQ PMBSL_AST+4
0116 368      ASSUME ASTPRM EQ ASTADR+4
50 10 AC 08 AC 7D 0116 369      MOVQ   ASTADR(AP), PMBSL_AST(R6); Save AST address and parameter
02 00 FEDC' 50 011B 370      EXTZV #0, #2, ACMODE(AP), R0  ; R0 <= request AST delivery mode
30 0121 371      BSBW   EXESMAXACMODE        ; Maximize requested and allowable
30 89 0124 372      BISB3 #<ACBSM NODELETE!ACB$M_PKAST>,- ; access modes & store with
2F A6 50 0126 373      MOVAB  R0, PMBSB_RMOD(R6)  ; nodelete, pkast set;quota, kast, clear
3C A6 02D1'CF 9E 0129 374      RSB    W^PFM_PKAST, PMBSL_KAST(R6); Set piggy-back kernel AST address
05 012F 375
0130 376
0130 377      :
0130 378      : Setup subprocess mode
0130 379      :
0130 380
0130 381      SET_SUBPMODE:          ; (R0-R3 DESTROYED)
0130 382
01  DD 0130 383      PUSHL  #SSS_NORMAL        ; Save room for status on stack
0132 384
0132 385      : Create a termination mailbox for the subprocess.
0132 386      :
0132 387      :
0132 388      SCREMBX_S CHAN=PMBSW_MBXCHN(R6), MAXMSG=#120,-
0132 389      BUFQUO=#120, PROMSK=#0
014E 390
6E 50 D0 014E 391      MOVL   R0, (SP)
03 50 E8 0151 392      BLBS   R0, $S
00D2 31 0154 393      BRW    PURGE_EXIT        ; exit on error
5E 10 C2 0157 394 5$:  SUBL2  #16, SP          ; buffer space for GETCHN on stack
5E DD 015A 395      PUSHL  SP          ; build descriptor for buffer
10 DD 015C 396      PUSHL  #16          ; length of buffer
52 6E DE 015E 397      MOVAL  (SP), R2        ; get descriptor address
0161 398      $GETCHN_S CHAN=PMBSW_MBXCHN(R6), PRIBUF=(R2); get mailbox unit #
0174 399
0174 400      :
0174 401      : Form unique subprocess name using EPID of this process
0174 402      : (NOTE: The following code adds a NET 4 bytes to the stack local storage)
0174 403      :
0174 404      PUSHR  #^M<R2, R3, R4, R5>

```

04 A2 FEC8 CF OF 28 0176 406 MOVC3 #15,W^FILWRT,4(R2) ; Move name template to stack storage
 3C BA 017D 407 POPR #^M<R2,R3,R4,R5>
 51 64 A4 D0 017F 408 MOVL PCBSL EPID(R4),R1 ; Convert hex EPID to ASCII
 57 0B A2 9E 0183 410 MOVAB 11(R2),R7 ; R1 <= EPID to be converted
 50 51 04 58 D0 0187 411 MOVL #28,R8 ; R7 <= address of 1st character
 87 00000000'9F40 90 018F 412 10\$: EXTZV R8,#4,R1,R0 ; R8 <= position of 1st nibble
 FFEC 58 FC 8F 00 9D 0197 413 MOVAB @#EXE\$AB HEXTAB[R0],(R7) ; Get next four bits of value
 62 04 A2 9E 019E 414 ACBB #0,-4,R8 10\$; Convert to ASCII and store
 OF DD 01A2 415 MOVAB 4(R2),(R2) ; Loop until all digits converted
 01A4 416 PUSHL #15 ; Form string descriptor
 01A4 417 : for resultant name
 01A4 418 : Create subprocess with high priority, full privilege and termination mailbox
 01A4 419 :
 01A4 420 :
 01A4 421 SCREPRC_S PIDADR=PMBSL EPID(R6),- ; pid of created process
 01A4 422 IMAGE=PFMFILWRT,-
 01A4 423 PRCNAM=-4(R2),-
 01A4 424 PRVADR=FILWRTPRV,-
 01A4 425 BASPRI=#6,-
 01A4 426 QUOTA=PFMQUOTA,-
 01A4 427 MBXUNT=8+12(R2),- ; unit from get channel information
 01A4 428 STSFLG=#4 ; disable swapping
 5E 1C C0 01CF 429 ADDL2 #<24+4>,SP ; Return stack local storage
 6E 50 D0 01D2 430 MOVL R0,(SP)
 03 50 E8 01D5 431 BLBS R0,15\$
 0082 31 01D8 432 BRW DA\$SGN_EXIT ; exit on error
 7E 54 7D 01DB 433 15\$: 434 MOVQ R4,-(SP) ; Save R4, R5
 50 34 A6 D0 01E2 435 SETIPL B^20\$; Synchronize access to system database
 00000000'EF 16 01E6 436 MOVL PMBSL EPID(R6),R0 ; Convert EPID to IPID
 30 A6 50 D0 01EC 437 JSB EXE\$EPID TO IPID
 00000000'EF 16 01FO 438 MOVL R0,PMBSL_PID(R6) ; and save in PMB
 011C C0 56 D0 01F6 439 JSB EXE\$IPID TO PCB ; Convert IPID to PCB address
 01FB 440 MOVL R6,PCBSL_PMB(R0) ; Insert PMB address in subprocess PCB
 54 8E 7D 01FE 441 SETIPL #0 ;
 8E D5 0201 442 MOVQ (SP)+,R4 ; Restore R4, R5
 FF07 31 0203 443 TSTL (SP)+ ; Discard stacked status
 0206 444 BRW START_OK ; and take successful exit path
 00000008 0206 445 20\$: .LONG IPL\$_SYNCH
 020A 446 :
 020A 447 : STOP - Termination Processing
 020A 448 :
 020A 449 :
 020A 450 :
 020A 451 STOP: 452 BSBW CHECK_ACMODE ; If caller does not have privilege.
 FE9F 30 020A 453 BLBS R0,5\$
 03 50 E8 020D 454 BRW NOPRIV ; return with error status
 FEAB 31 0210 455 5\$: BBCC #PHDSV_PFMFLG,PHDSW_FLAGS(R5),10\$; Turn off active monitoring
 00 36 A5 00 E5 0213 456 10\$: JSB FLUSH_BUFFER ; Flush outstanding buffers
 00000065'EF 16 0218 457 021E ; (R2, R4, R5 DESTROYED)
 021E 458 SETIPL #0 ; return to IPL 0
 01 DD 0221 459 PUSHL #SS\$ NORMAL ; Assume success
 0223 460 ASSUME PMBSV_MODE EQ 0
 0223 461 ASSUME PMBSK_IMAGE EQ 1
 02 0B A6 E8 0223 462 BLBS PMBSB_FLAGS(R6),PURGE_EXIT ; If subprocess mode,

06 11 0227 463 BRB STOP_PROCESS : do orderly shutdown of subprocess
0229 464 PURGE_EXIT: ; (Assumes IPL=0, (SP) = status)
50 8ED0 0229 465 POPL R0
022C 466 PURGE_EXIT R0:
3C 10 022C 467 BSBB PFMSPURGE ; Return process buffers to system
04 022E 468 RET ; Return w/status in R0
022F 469
022F 470:
022F 471: Orderly shutdown of subprocess
022F 472:
022F 473:
022F 474 STOP_PROCESS:
1D 50 E9 022F 475 SFORCEX_S PIDADR=PMB\$L_EPID(R6) ; force subprocess to exit
023D 476 BLBC R0,DASSGN_EXIT ; don't wait for message if force failed
0240 477 \$QIOW_S CHAN=PMB\$W_MBXCHN(R6),FUNC=#IOS_READVBLK,-
0240 478 P1=PMB\$L_EPID(R6),- ; buffer four bytes of termination
0240 479 P2=#4 ; ...message in unneeded PID slot
025D 480 DASSGN_EXIT:
025D 481 \$DASSGN_S CHAN=PMB\$W_MBXCHN(R6) ; release mailbox
BF 11 0268 482 BRB PURGE_EXIT
026A 483

026A 485 .SBTTL PFM\$PURGE - Return all process buffers to pool
 026A 486
 026A 487 :++
 026A 488
 026A 489 : FUNCTIONAL DESCRIPTION:
 026A 490
 026A 491 : All data buffers and the control block are returned to pool.
 026A 492
 026A 493 : CALLING SEQUENCE:
 026A 494
 026A 495 : BSB/JSB PFM\$PURGE
 026A 496
 026A 497 : INPUTS:
 026A 498
 026A 499 : none
 026A 500
 026A 501 : IMPLICIT INPUTS:
 026A 502
 026A 503 : PCB Address
 026A 504
 026A 505 : OUTPUTS:
 026A 506
 026A 507 : none
 026A 508
 026A 509 : IMPLICIT OUTPUTS:
 026A 510
 026A 511 : none
 026A 512
 026A 513 : SIDE EFFECTS:
 026A 514
 026A 515 : none - all registers preserved
 026A 516
 026A 517 : ROUTINE VALUE:
 026A 518
 026A 519 : none
 026A 520
 026A 521 : ENVIRONMENT:
 026A 522
 026A 523 : Kernel mode, IPL <= IPL\$_ASTDEL
 026A 524 :--
 026A 525
 026A 526
 026A 527 PFM\$PURGE::
 54 007F 8F BB 026A 528 PUSHR #^M<R0,R1,R2,R3,R4,R5,R6> ; save registers
 00000000'EF D0 026E 529 MOVL L^SCH\$GL_CURPCB,R4 ; get our PCB address
 0275 530 DSBINT #IPL\$_ASTDEL ; protect manipulation of PMB
 56 011C C4 D0 027B 531 MOVL PCB\$L_PMB(R4),R6 ; get PMB address (if any)
 011C C4 D4 0280 532 CLRL PCB\$L_PMB(R4) ; clear PMB pointer
 56 D5 0284 533 TSTL R6 ; was there a PMB
 37 13 0286 534 BEQL 40\$; exit if not
 02 0B A6 02 E5 0288 535 BBCC #PMB\$V_QAST,PMB\$B_FLAGS(R6),5\$; Is ACB enqueued on PCB ?
 47 10 028D 536 BSBB DEQUEUE_ACB ; Yes, remove it.
 66 D5 028F 537 5\$: TSTL PMB\$L_CURBUF(R6) ; check if current buffer there
 06 13 0291 538 BEQL 10\$; branch if none there
 50 04 A6 D0 0293 539 MOVL PMB\$L_BUFBASE(R6),R0 ; else set buffer address
 2E 10 0297 540 BSBB 50\$; return to system
 50 14 B6 0F 0299 541 10\$: REMQUE @PMB\$Q_HDR(R6),R0 ; remove buffer from queue

04 1D 029D 542 BVS 20\$; exit if none there
 26 10 029F 543 BSB 50\$; return buffer
 F6 11 02A1 544 BRB 10\$; go back for more buffers
 04 1D 02A3 545 20\$: REMQUE @PMB\$Q_SBPHDR(R6),R0 ; check that subprocess queue is clear
 04 10 02A7 546 BVS 30\$; exit if no entry
 1C 10 02A9 547 BSB 50\$; else return buffer to system pool
 F6 11 02AB 548 BRB 20\$; check for any more
 50 56 D0 02AD 549 30\$: MOVL R6,R0 ; get back PMB block address
 15 10 02B0 550 BSB 50\$; deallocate control block
 20 A0 50 0080 C4 D0 02B2 551 MOVL PCB\$L_JIB(R4),R0 ; Get JIB address
 00000A7C 8F C0 02B7 552 ADDL2 #QUOTA_CHARGE,JIB\$L_BYTCNT(R0) ; Return quota to process
 007F 8F BA 02C2 553 40\$: ENBINT ; restore IPL
 05 02C6 554 POPR #^M<R0,R1,R2,R3,R4,R5,R6> ; restore registers
 02C7 555 RSB ; return to caller
 02C7 556
 02C7 557
 02C7 558 ; Return buffer to pool.
 02C7 559
 02C7 560
 0A A0 00000000'EF B4 02C7 561 50\$: CLRW PMB\$W_SIZE+2(R0) ; Clear type field for EXE\$DEANONPAGED
 16 02CA 562 JSB EXE\$DEANONPAGED ; deallocate memory
 05 02D0 563 RSB
 02D1 564
 02D1 565
 02D1 566 ; The sole purpose of this routine is to mark the ACB as not queued.
 02D1 567 ; It is called as a piggy-back kernel AST routine to effectively interlock the
 02D1 568 ; operation with the monitoring termination code, which will conditionally
 02D1 569 ; remove the ACB from the PCB.
 02D1 570
 02D1 571 ; Inputs: R5 = ACB address, IPL = IPL\$_ASTDEL
 02D1 572 ; Outputs: PMB\$V_QAST cleared, all registers preserved
 02D1 573
 02D1 574
 02D1 575 PFM_PKAST:
 E7 A5 04 8A 02D1 576 BICB2 #PMB\$M_QAST,PMB\$B_FLAGS-PMB\$L_ASTQFL(R5) ; Show ACB dequeued
 05 02D5 577 RSB
 02D6 578
 02D6 579
 02D6 580 ; This routine is called from PFM\$PURGE to dequeue the ACB
 02D6 581 ; if PMB\$V_QAST in PMB\$B_FLAGS was set.
 02D6 582
 02D6 583 ; Inputs: R6 = PMB address, IPL=IPL\$_ASTDEL,
 02D6 584 ; PMB\$V_QAST checked and cleared by caller
 02D6 585 ; Outputs: ACB dequeued, R5 destroyed
 02D6 586
 02D6 587
 02D6 588 DEQUEUE_ACB:
 55 24 A6 9E 02D6 589 MOVAB PMB\$L_ASTQFL(R6),R5 ; R5 <= address of embedded ACB
 00000000'EF 16 02DA 590 SETIPL B^10\$
 02DE 591 JSB SCH\$REMOVACB ; Remove ACB from PCB
 02E4 592 SETIPL #IPL\$_ASTDEL
 05 02E7 593 RSB
 00000008 02E8 594 10\$: .LONG IPL\$_SYNCH
 02EC 595

02EC 597 .SBTTL PFM\$GETBUF - Return PFM Buffer to Caller
02EC 598
02EC 599 :++
02EC 600
02EC 601 : FUNCTIONAL DESCRIPTION:
02EC 602
02EC 603 Returns a filled PFM buffer to the caller.
02EC 604 For efficiency, this routine and FLUSH_BUFFER assume
02EC 605 that once the collection process is awakened, it will
02EC 606 continue calling PFM\$GETBUF until SSS_NODATA is returned, or
02EC 607 an error is encountered.
02EC 608
02EC 609 : CALLING SEQUENCE:
02EC 610
02EC 611 BSB/JSB PFM\$GETBUF
02EC 612
02EC 613 : INPUTS:
02EC 614
02EC 615 R1 = Buffer address
02EC 616 R2 = Buffer size
02EC 617 (Should be = PFBSS_USER_BUFFER, with which it is minimized)
02EC 618
02EC 619 NB: No checks are made for buffer accessibility.
02EC 620
02EC 621 : IMPLICIT INPUTS:
02EC 622
02EC 623 SChSGL_CURPCB - PCB address of current process
02EC 624 PCB\$L_PMB in PCB - pointer to PMB
02EC 625
02EC 626 : OUTPUTS:
02EC 627
02EC 628 If R0=SSS_NORMAL or SSS_BUFFEROVF, buffer is filled with page fault data
02EC 629
02EC 630 : IMPLICIT OUTPUTS:
02EC 631
02EC 632 none
02EC 633
02EC 634 : ROUTINE VALUE:
02EC 635
02EC 636 R0 = Status
02EC 637
02EC 638 SSS_NORMAL
02EC 639 SSS_NODATA
02EC 640 SSS_ILLSEQOP
02EC 641 SSS_BUFFEROVF
02EC 642
02EC 643 : SIDE EFFECTS:
02EC 644
02EC 645 R1-R5 destroyed
02EC 646
02EC 647 : ENVIRONMENT:
02EC 648
02EC 649 Kernel mode, IPL 0
02EC 650 :--
02EC 651
02EC 652 : NB: This code assumes that IPL synchronization (above ASTDEL) is
02EC 653 : not required because

02EC 654 : a) a command/response protocol is used between the
 02EC 655 : main and sub processes
 02EC 656 : b) SYSRUNDWN will unconditionally call SETPFM to
 02EC 657 : turn off monitoring (i.e., assumption a) cannot be breached)
 02EC 658 : c) no code outside of this module accesses PFM data structures
 02EC 659 :
 02EC 660 PFMSGETBUF::
 54 00000000'EF D0 02EC 661 MOVL L^SCH\$GL_CURPCB,R4 ; Get PCB address for this process
 54 011C C4 D0 02F3 662 MOVL PCB\$L_PMB(R4),R4 ; Get PMB address
 3A 13 02F8 663 BEQL 50\$; BR if none
 55 1C B4 0F 02FD 664 10\$: SETIPL #IPL\$_ASTDEL ; Protect buffer handling
 26 1D 0301 665 REMQUE @PMB\$0_SBPHDR(R4),R5 ; Dequeue filled buffer
 53 50 01 3C 0303 666 BVS 40\$; Exit if none
 53 0200 8F 3C 0306 667 MOVZWL #SS\$_NORMAL, R0 ; Assume adequate buffer size
 53 52 D1 030B 668 MOVZWL #PFB\$S_USER_BUFFER, R3 ; and set number of bytes to move
 08 18 030E 669 CMPL R2, R3 ; Is buffer big enough ?
 50 0601 8F 3C 0310 670 BGEQ 15\$; BR if yes
 53 52 D0 0315 671 MOVZWL #SS\$_BUFFEROVF, R0 ; Indicate data loss
 31 BB 0318 672 MOVL R2, R3 ; Adjust number of bytes to move
 61 0C A5 53 28 031A 673 15\$: PUSHR #^M<R0, R4, R5> ; Save regs and status
 31 BA 031F 674 MOVC3 R3, PFB\$B_USER_BUFFER(R5), (R1) ; Move buffer
 14 A4 65 0E 0321 675 POPR #^M<R0, R4, R5> ; Restore regs and status
 05 0325 676 INSQUE (R5), PMBSQ_HDR(R4) ; Return buffer to free list
 05 0328 677 20\$: SETIPL #0 ; Restore IPL
 0329 678 30\$: RSB ; Return to caller
 50 01AC 8F 3C 0329 680 40\$: MOVZWL #SS\$_NODATA, R0
 0B A4 02 8A 032E 681 BICB2 #PMB\$M_ASTIP, PMBSB_FLAGS(R4) ; Show no AST in progress
 F1 11 0332 682 BRB 20\$
 50 02DC 8F 3C 0334 683 50\$: MOVZWL #SS\$_ILLSEQOP, R0
 ED 11 0339 684 BRB 30\$
 033B 685

033B 687 .SBTTL ALLPMB - Allocate PMB control block and data buffers
033B 688
033B 689 :++
033B 690 :
033B 691 : FUNCTIONAL DESCRIPTION:
033B 692 :
033B 693 : Allocates all process structures needed for page fault monitoring.
033B 694 :
033B 695 : CALLING SEQUENCE:
033B 696 :
033B 697 : JSB/BSB ALLPMB
033B 698 :
033B 699 : INPUTS:
033B 700 :
033B 701 : R4 PCB address
033B 702 : R5 PHD address
033B 703 :
033B 704 :
033B 705 : IMPLICIT INPUTS:
033B 706 :
033B 707 : none
033B 708 :
033B 709 : OUTPUTS:
033B 710 :
033B 711 : R6 = PMB address
033B 712 :
033B 713 : IMPLICIT OUTPUTS:
033B 714 :
033B 715 : PCB\$L_PMB contains PMB address
033B 716 :
033B 717 : ROUTINE VALUE:
033B 718 :
033B 719 : R0 = SSS_NORMAL
033B 720 : = Pool allocation error (SSS_INSFMEM, etc.)
033B 721 :
033B 722 : SIDE EFFECTS:
033B 723 :
033B 724 : R0,R1,R2,R3,R8,R9 destroyed
033B 725 :
033B 726 : ENVIRONMENT:
033B 727 :
033B 728 : Kernel mode, IPL = IPL\$_ASTDEL
033B 729 :--
033B 730 :
033B 731 : ALLPMB:
1C DD 033B 732 PUSHL #SSS_EXQUOTA ; stack no quota error code
033D 733
033D 734 :
033D 735 : Process should have enough quota for data buffers and PMB
033D 736 :
033D 737 :
033D 738 MOVL PCB\$L_JIB(R4),R8 ; get JIB address
033D 739 CMPL #QUOTA_CHARGE,JIB\$L_BYTCNT(R8) ; quota left?
0342 740 BGTRU 20\$; error if not
034A 741
034C 742 :
034C 743 : Allocate PMB control block and insert address in PMB list

20 AB 58 0080 C4 DD
00000A7C 8F D1
6F 1A 0342
034A
034C
034C

```

034C 744 ;  

034C 745 ;  

6E 0124 BF 3C 034C 746 ;  

51 0040 BF 3C 0351 747 ;  

00000000'EF 16 0356 748 ;  

5C 50 E9 035C 749 ;  

20 AB 00000A7C 8F C2 035F 750 ;  

56 52 D0 0367 751 ;  

011C C4 56 D0 036A 752 ;  

036F 753 ;  

036F 754 ;  

036F 755 ;  

036F 756 : Initialize PMB and allocate and queue data buffers.  

036F 757 ;  

036F 758 ;  

036F 759 ;  

0A A6 46 8F 90 036F 760 ;  

0374 761 ;  

0374 762 ;  

0374 763 ;  

0374 764 ;  

50 14 A6 9E 0374 765 ;  

60 60 DE 0378 766 ;  

80 80 DE 037B 767 ;  

60 60 DE 037E 768 ;  

80 80 DE 0381 769 ;  

60 60 DE 0384 770 ;  

60 80 DE 0387 771 ;  

038A 772 ;  

66 7C 038A 773 ;  

038C 774 ;  

038C 775 ;  

0B A6 94 038C 776 ;  

10 A6 D4 038F 777 ;  

30 A6 60 A4 D0 0392 778 ;  

0C A6 01 CE 0397 779 ;  

0398 780 ;  

59 05 3C 039B 781 ;  

51 020C 8F 3C 039E 782 10$: ;  

00000000'EF 16 03A3 783 ;  

OF 50 E9 03A9 784 ;  

0A A2 47 8F 90 03AC 785 ;  

14 A6 62 0E 03B1 786 ;  

E6 59 F5 03B5 787 ;  

03B8 788 ;  

6E 01 3C 03B8 789 ;  

50 8E D0 03BB 790 20$: ;  

05 03BE 791 ;  

03BF 792 ;  

034C 746 ;  

034C 747 ;  

MOVZWL #SSS_INSFMEM,(SP) ; assume memory not available  

MOVZWL #PMB$C_LENGTH,R1 ; get length of PMB block to allocate  

JSB EXE$ALLOCBUF ; allocate control block  

BLBC R0,20$ ; check that memory available  

SUBL2 #QUOTA_CHARGE,JIBSL_BYTCNT(R8) ; adjust quota  

MOVL R2,R6 ; copy PMB block address  

MOVL R6,PCB$L_PMB(R4) ; insert PMB address in PCB  

; N.B.: PFM$PURGE assumes that quota has  

; been charged if PCB$L_PMB <> 0  

MOVBL #DYNSC_PMB,PMB$B_TYPE(R6) ; Set structure type to PMB  

ASSUME PMB$Q_SBPHDR EQ PMB$Q_HDR+8  

ASSUME PMB$L_ASTQFL EQ PMB$Q_SBPHDR+8  

MOVAB PMB$Q_HDR(R6),R0 ; get queue header address  

MOVAL (R0),T(R0) ; init empty queue flink  

MOVAL (R0)+,(R0)+ ; init empty queue blink  

MOVAL (R0),(R0) ; init subprocess queue flink  

MOVAL (R0)+,(R0)+ ; init subprocess queue blink  

MOVAL (R0),(R0) ; init AST queue flink  

MOVAL (R0)+,(R0) ; init AST queue blink  

ASSUME PMB$L_BUFBASE EQ PMB$L_CURBUF+4  

CLRQ PMB$L_CURBUF(R6) ; note no current buffer  

CLRB PMB$B_FLAGS(R6) ; insure that all flags are clear  

CLRL PMB$L_OVERFLOW(R6) ; indicate no overflows  

MOVL PCB$L_PID(R4),PMB$L_PID(R6) ; Insert PID into PMB/ACB  

MNEG L #1,PMB$L_LASTCPU(R6) ; force timestamping of 1st record  

MOVZWL #BUFCNT,R9 ; number of data buffers to allocate  

MOVZWL #BUFSIZ,R1 ; get size of buffer to allocate  

JSB EXE$ALLOCBUF ; allocate buffer  

BLBC R0,20$ ; take error path if no memory available  

MOVB #DYNSC_PFB,PFB$B_TYPE(R2) ; Set structure type to PFB  

INSQUE (R2),PMB$Q_HDR(R6) ; queue on empty buffer list  

SOBGTR R9,10$ ; back for more buffers  

MOVZWL #SSS_NORMAL,(SP) ; Indicate success  

POPL R0 ; Return w/status in R0  

RSB

```

03BF 794 .SBTTL PFMSMON - Resident Monitoring Code
 03BF 795
 03BF 796 :++
 03BF 797
 03BF 798 : FUNCTIONAL DESCRIPTION:
 03BF 799
 03BF 800 : Resident code called by memory management to record
 03BF 801 : page fault PC and VA. Data is inserted into a buffer.
 03BF 802 : When the buffer is full, it is queued for a cooperating
 03BF 803 : process which outputs the data to disk.
 03BF 804
 03BF 805 : CALLING SEQUENCE:
 03BF 806
 03BF 807 : BSB/JSB PFMSMON
 03BF 808
 03BF 809 : INPUTS:
 03BF 810
 03BF 811 : R4 = PCB address
 03BF 812 : R5 = PHD address
 03BF 813
 03BF 814 : IMPLICIT INPUTS:
 03BF 815
 03BF 816 : none
 03BF 817
 03BF 818 : OUTPUTS:
 03BF 819
 03BF 820 : none
 03BF 821
 03BF 822 : IMPLICIT OUTPUTS:
 03BF 823
 03BF 824 : none
 03BF 825
 03BF 826 : ROUTINE VALUE:
 03BF 827
 03BF 828 : none
 03BF 829
 03BF 830 : SIDE EFFECTS:
 03BF 831
 03BF 832 : none
 03BF 833
 03BF 834 : ENVIRONMENT:
 03BF 835
 03BF 836 : Kernel mode, IPL = IPL\$_SYNCH
 03BF 837 :--
 03BF 838
 03BF 839
 00000000 840 .PSECT AEXENONPAGED, LONG
 0000 841
 0000 842 : ENABLE LSB
 02 0C A2 D1 0000 843 5\$: CMPL PFBSL_RECCNT(R2), #2 : Space for time stamp and PC/VA pair ?
 0C 1E 0004 844 BGEQU 6\$: BR if space in buffer
 55 DD 0006 845 PUSHL R5 : Save PHD address
 61 10 0008 846 BSBB FLUSH_BUFFER_INT : Flush buffer (R2, R4, R5 DESTROYED)
 55 8ED0 000A 847 POPL R5 : Restore PHD address
 0093 30 000D 848 BSBW GETBUF : try to get next buffer
 49 1D 0010 849 BVS 30\$: exit if none there, lose data
 81 01 CE 0012 850 6\$: MNEGL #1, (R1)+ : Add time stamp to buffer

```

81 38 A5 D0 0015 851 SETIPL #IPL$_HWCLK ; Synchronize with CPU time updating
38 A5 D0 0018 852 MOVL PHDSL_CPUTIM(R5),(R1)+ ; Update reference CPU time
DC A6 D0 001C 853 MOVL PHDSL_CPUTIM(R5)- PMBSL_LASTCPU(R6)
0C A2 D7 0021 854 SETIPL #IPL$_SYNCH ; Lower IPL to entry IPL
1C 11 0024 855 DECL PFBSL_RECCNT(R2) ; Adjust record count for time stamp
0027 856 BRB 15$ ; Rejoin mainline code
0029 857
0029 858
002C 859 .ALIGN LONG
PFM$MON: 860
56 007F 8F BB 002C 861 PUSHR #^M<R0,R1,R2,R3,R4,R5,R6> ; save registers
011C C4 D0 0030 862 MOVL PCBSL_PMB(R4),R6 ; get PMB address
0035 863
0035 864 ASSUME PMBSL_BUFBASE EQ PMBSL_CURBUF+4
0035 865
51 66 7D 0035 866 MOVQ PMBSL_CURBUF(R6),R1 ; get buffer address & base
04 12 0038 867 BNEQ 10$ ; branch if buffer exists
67 10 003A 868 BSBB GETBUF ; try to get next buffer
1D 1D 003C 869 BVS 30$ ; exit if none there, lose data
OC A6 38 A5 D1 003E 870 10$: CMPL PHDSL_CPUTIM(R5),PMBSL_LASTCPU(R6) ; Need a CPU time stamp ?
BB 12 0043 871 BNEQ 5$ ; BR if yes
81 40 AE D0 0045 872 15$: MOVL FAULTPC(SP),(R1)+ ; insert pc of instruction
81 3C AE D0 0049 873 MOVL FAULTVA(SP),(R1)+ ; insert va which faulted
66 51 D0 004D 874 MOVL R1,PMBSL_CURBUF(R6) ; save current buffer address
02 0C A2 F5 0050 875 SOBGTR PFBSL_RECCNT(R2),20$ ; Buffer full ?
15 10 0054 876 BSBB FLUSH_BUFFER_INT ; Yes, flush it
0056 877
0056 878 20$: POPR #^M<R0,R1,R2,R3,R4,R5,R6> ; restore registers
05 005A 879 RSB
10 A6 D6 005B 880 30$: INCL PMBSL_OVERFLOW(R6) ; Count an overflow
F6 11 005E 881 BRB 20$ ; .DISABLE LSB
0060 882
0060 883
0060 884 .ENABLE LSB
0060 885
0060 886 5$: MOVL #MAXREC,PFBSL_RECCNT(R2) ; Re-initialize records remaining
05 0064 887 RSB ; and return
0065 888
0065 889 FLUSH_BUFFER: ; (R2, R4, R5 DESTROYED)
0065 890 MOVL PMBSL_BUFBASE(R6),R2 ; Entry with IPL=IPL$_ASTDEL
25 13 0069 891 BEQL 20$ ; Get current buffer base address
3F 0C A2 C3 006B 892 FLUSH_BUFFER_INT: ; Just exit if no current buffer
OC A2 006B 893 SUBL3 PFBSL_RECCNT(R2),#MAXREC,- ; Entry with IPL=IPL$_SYNCH, R2 set
ED 13 0071 894 PFBSL_RECCNT(R2) ; Convert records Remaining to
0073 895 BEQL 5$ ; number of records in buffer
20 B6 62 0E 0073 896 ASSUME PFBSL_FLINK EQ 0 ; BR if no records in buffer
0077 897 INSQUE (R2),#PMB$Q_SBPHDR+4(R6) ; insert buffer at end of write queue
0077 898 ASSUME PMBSV_MODE EQ 0
0077 900 ASSUME PMBSK_IMAGE EQ 1
16 0B A6 E9 0077 901 BLBC PMBSB_FLAGS(R6),30$ ; BR if subprocess mode
0E 0B A6 01 E2 007B 902 BBSS #PMB$V_ASTIP,PMBSB_FLAGS(R6),10$ ; If AST in progress, return
0B A6 04 88 0080 903 BISB2 #PMB$M_QAST,PMBSB_FLAGS(R6) ; Show embedded ACB queued on PCB
52 01 9A 0084 904 MOVZBL #PRIS_TOCOM,R2 ; R2 <= priority increment class
55 24 A6 DE 0087 905 MOVAL PMBSL_ASTQFL(R6),R5 ; R5 <= address of ACB
FF72 30 008B 906 BSBW SCHSQAST ; Signal full buffer available
66 7C 008E 907 10$: CLRQ PMBSL_CURBUF(R6) ; Note that no current buffer exists

```

05 0090 908 20\$: RSB

51 30 A6 D0 0091 910 30\$: MOVL PMB\$L_PID(R6),R1 ; get params to wake up other process
0095 911 DSBINT #IPL\$-SYNCH
FF62' 30 009B 912 BSBW SCH\$WAKE ; and wake the process
EB 11 00A1 914 ENBINT
00A3 915 BRB 10\$
00A3 916 .DISABLE LSB
00A3 917
52 14 B6 0F 00A3 918 GETBUF: 919 REMQUE @PMB\$Q_HDR(R6),R2 ; get next buffer from queue
14 1D 00A7 920 BVS 10\$; dequeue next buffer
04 A6 52 D0 00A9 921 MOVL R2,PMB\$L_BUFBASE(R6) ; leave with V set if none there
51 14 A2 9E 00AD 922 MOVAB PFB\$B_BUFFER(R2),R1 ; save buffer base address in PMB
0C A2 3F D0 00B1 923 MOVL #MAXREC,PFB\$L_REC_CNT(R2) ; skip buffer overhead
10 A2 10 A6 D0 00B5 924 MOVL PMB\$L_OVERFLOW(R6),PFB\$L_OVERFLOW(R2) ; Initialize records remaining
10 A6 D4 00BA 925 CLRL PMB\$L_OVERFLOW(R6) ; Copy overflow count
00BD 926 ; and then reset it (also clears V)
00BD 927 10\$: RSB ; (R1 & R2 point to buffer)
00BE 928 ; return with V set or clear
00BE 929 .END

| | | | | |
|-------------------|-----------------|---|-------------------|---------------|
| SST1 | = 00000001 | | PFMQUOTA | 00000020 R 02 |
| ACB\$M_NODELETE | = 00000020 | | PFM_PKAST | 000002D1 R 02 |
| ACB\$M_PKAST | = 00000010 | | PHDSL_CPUTIM | = 00000038 |
| AC\$MODE | = 00000010 | | PHDSV_PFMFLG | = 00000000 |
| ALLPMB | = 0000033B R 02 | | PHDSW_FLAGS | = 00000036 |
| ASTADR | = 00000008 | | PMB\$B_AC\$MODE | = 0000002E |
| ASTPRM | = 0000000C | | PMB\$B_FLAGS | = 00000008 |
| BUFCNT | = 00000005 | | PMB\$B_RMOD | = 0000002F |
| BUFSIZ | = 0000020C | | PMB\$B_TYPE | = 0000000A |
| CHECK_AC\$MODE | 000000AC R 02 | | PMB\$C_LENGTH | = 00000040 |
| CHECK_STOP | 000000CE R 02 | | PMB\$K_IMAGE | = 00000001 |
| CTL\$GC_PHD | ***** X 02 | | PMB\$L_AST | = 00000034 |
| DASSGN_EXIT | 0000025D R 02 | | PMB\$L_ASTPRM | = 00000038 |
| DEQUEUE_ACB | 000002D6 R 02 | | PMB\$L_ASTQFL | = 00000024 |
| DYN\$C_PFB | = 00000047 | | PMB\$L_BUFBASE | = 00000004 |
| DYN\$C_PMB | = 00000046 | | PMB\$L_CURBUF | = 00000000 |
| EXE\$AB_HEXTAB | ***** X 02 | | PMB\$L_EPID | = 00000034 |
| EXE\$AL\$OCBUF | ***** X 02 | | PMB\$L_KAST | = 0000003C |
| EXE\$DEANONPAGED | ***** X 02 | | PMB\$L_LASTCPU | = 0000000C |
| EXE\$EPID_TO_IPID | ***** X 02 | | PMB\$L_OVERFLOW | = 00000010 |
| EXE\$IPID_TO_PCB | ***** X 02 | | PMB\$L_PID | = 00000030 |
| EXE\$MAXAC\$MODE | ***** X 02 | | PMB\$M_ASTIP | = 00000002 |
| EXE\$SETPFM | 0000006A RG 02 | | PMB\$M_QAST | = 00000004 |
| FAULTPC | = 00000040 | | PMB\$Q_HDR | = 00000014 |
| FAULTVA | = 0000003C | | PMB\$Q_SBPHDR | = 0000001C |
| FILWRT | = 00000043 R 02 | | PMB\$V_ASTIP | = 00000001 |
| FILWRTPRV | 00000052 R 02 | | PMB\$V_MODE | = 00000000 |
| FLUSH_BUFFER | 00000065 R 03 | | PMB\$V_QAST | = 00000002 |
| FLUSH_BUFFER_INT | 0000006B R 03 | | FMBSW_MB\$XCHN | = 0000002C |
| GETBUF | 000000A3 R 03 | | PMB\$W_SIZE | = 00000008 |
| ILLSEQOP | 00000061 R 02 | | PQLS_BYTLM | = 00000003 |
| IOS_READVBLK | = 00000031 | | PQLS_CPLUM | = 00000004 |
| IPL\$_ASTDEL | = 00000002 | | PQLS_FILLM | = 00000006 |
| IPL\$_HWCLK | = 00000018 | | PQLS_LISTEND | = 00000000 |
| IPL\$_SYNCH | = 00000008 | | PQLS_PGF\$QUOTA | = 00000007 |
| JIBS[_BYTCNT] | = 00000020 | | PQLS_PRCLM | = 00000008 |
| MAXREC | = 0000003F | | PQLS_TQELM | = 00000009 |
| NOPRIV | 000000BE R 02 | | PR\$_IPL | = 00000012 |
| PCBSL_EPID | = 00000064 | | PR\$5_IOCOM | = 00000001 |
| PCBSL_JIB | = 00000080 | | PSL\$5_PRV\$MOD | = 00000002 |
| PCBSL_PID | = 00000060 | | PSL\$V_PRV\$MOD | = 00000016 |
| PCBSL_PMB | = 0000011C | | PURGE_EXIT | 00000229 R 02 |
| PFBSB_BUFFER | = 00000014 | | PURGE_EXIT_RO | 0000022C R 02 |
| PFBSB_TYPE | = 0000000A | | PURGE_EXIT_RO_VEC | 000000C8 R 02 |
| PFBSB_USER_BUFFER | = 0000000C | | PURGE_EXIT_VEC | 000000CB R 02 |
| PFBS\$C_LENGTH | = 0000020C | | QUOTA_CHARGE | = 00000A7C |
| PFBS\$FLINK | = 00000000 | | SCHSG\$CURPCB | ***** X 02 |
| PFBS\$OVERFLOW | = 00000010 | | SCHSQAST | ***** X 03 |
| PFBS\$RECCNT | = 0000000C | | SCH\$REMOVACB | ***** X 02 |
| PFBS\$USER_BUFFER | = 00000200 | | SCH\$WAKE | ***** X 03 |
| PFM\$C_BUFCNT | = 00000005 | G | SET_ASTMODE | 00000116 R 02 |
| PFM\$GETBUF | 000002EC RG 02 | | SET_SUBPMODE | 00000130 R 02 |
| PFM\$MON | 0000002C RG 03 | | SS\$_BUFFEROVF | = 00000601 |
| PFM\$PURGE | 0000026A RG 02 | | SS\$_EXQUOTA | = 0000001C |
| PFM\$BUSY | 0000005A R 02 | | SS\$_ILLSEQOP | = 000002DC |
| PFM\$FLWRT | 00000000 R 02 | | SS\$_IN\$FMEM | = 00000124 |
| PFM\$FLG | = 00000004 | | SS\$_NODATA | = 000001AC |

```

SS$_NOPRIV      = 00000024
SS$_NORMAL      = 00000001
SS$_PFMBSY      = 00000204
START           000000E3 R  02
START_OK         0000010D R  02
STOP             0000020A R  02
STOP_PROCESS    0000022F R  02
STOP_VEC         000000C5 R  02
SUBFUNC          00000090 R  02
SYSSCREMBX      ***** GX  02
SYSSCREPRC      ***** GX  02
SYSSDASSGN      ***** GX  02
SYSSFORCEX      ***** GX  02
SYSSGETCHN      ***** GX  02
SYSSQIOW         ***** GX  02

```

```

+-----+
! Psect synopsis !
+-----+

```

| PSECT name | Allocation | PSECT No. | Attributes |
|--------------|------------------|-----------|---|
| ABS | 00000000 (0.) | 00 (0.) | NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE |
| \$ABSS | 00000000 (0.) | 01 (1.) | NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE |
| YEXEPAGED | 000003BF (959.) | 02 (2.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG |
| AEXENONPAGED | 000000BE (190.) | 03 (3.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG |

```

+-----+
! Performance indicators !
+-----+

```

| Phase | Page faults | CPU Time | Elapsed Time |
|------------------------|-------------|-------------|--------------|
| Initialization | 29 | 00:00:00.07 | 00:00:00.26 |
| Command processing | 106 | 00:00:00.55 | 00:00:01.73 |
| Pass 1 | 407 | 00:00:15.76 | 00:00:35.84 |
| Symbol table sort | 0 | 00:00:02.35 | 00:00:05.76 |
| Pass 2 | 165 | 00:00:03.46 | 00:00:08.12 |
| Symbol table output | 16 | 00:00:00.14 | 00:00:00.22 |
| Psect synopsis output | 2 | 00:00:00.02 | 00:00:00.02 |
| Cross-reference output | 0 | 00:00:00.00 | 00:00:00.00 |
| Assembler run totals | 727 | 00:00:22.36 | 00:00:51.95 |

The working set limit was 1500 pages.

88818 bytes (174 pages) of virtual memory were used to buffer the intermediate code.

There were 80 pages of symbol table space allocated to hold 1517 non-local and 40 local symbols.

929 source lines were read in Pass 1, producing 20 object records in Pass 2.

38 pages of virtual memory were used to define 36 macros.

! Macro library statistics !

Macro library name

-\$255\$DUA2B:[SYS.OBJ]LIB.MLB;1
-\$255\$DUA2B:[SYSLIB]STARLET.MLB;2
TOTALS (all libraries)

Macros defined

13
19
32

1718 GETS were required to define 32 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LISS:SYSSETPFM/OBJ=OBJ\$:SYSSETPFM MSRC\$:SYSSETPFM/UPDATE=(ENH\$:SYSSETPFM)+EXECMLS/LIB

0388 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

